Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Отчёт

Параллельная реализация решения СЛАУ с помощью метода отражений.

Работу выполнил:
Лесцов Б.А.
423 группа

2018

Постановка задачи и формат данных.

Задача: Реализовать параллельный алгоритм решения СЛАУ с помощью метода отражений.

Постановка задачи:

Заданы невырожденная вещественная матрица *A* размера *NxN* и вещественный вектор *b* длины N такие что система *Ax=b* имеет единственное решение. Необходимо найти вектор *x,* являющийся решением этой системы.

Компиляция:

1) С помощью gnu make:

> make

2) С помощью cmake:

> mkdir build

> cd build

> cmake ..

Запуск:

1) Считать матрицу из файла:

> mpirun -np ‘число процессов’ build/bin/Prak ‘путь к файлу с матрицей’

Пример*:*

> mpirun -np 2 build/bin/Prak data/mat.txt

2) Сгенерировать с помощью функции f(i, j), заданной в source/main.cpp:

> mpirun -np ‘число процессов’ build/bin/Prak ‘N’ ‘N+1’, где N – размер матрицы системы

Пример*:*

> mpirun -np 4 build/bin/Prak 1024 1025

Формат входных файлов:

Входной файл:

1) Файл с матрицей размера *Nx(N+1)* в текстовом виде. В начале файла располагаются два числа m, n типа size_t – размеры матрицы. Далее следуют n*m вещественных чисел – сама матрица. Последний столбец этой матрицы – это вектор *b.*

Формат входных файлов:
В стандартный поток вывода будет выдано решение заданной СЛАУ в формате:

x_1 = ‘значение переменной x_1’
...
x_i = ‘значение переменной x_i’
...
x_N = ‘значение переменной x_N’

Далее следует строка:

Mat_size ‘размер матрицы’ Comm_size ‘число процессов’
Forward_Time_(microsec) ‘время приведения к верхнетреугольному виду в микросекундах’
Backward_Time_(microsec) ‘время обратного хода метода Гаусса’
diff ‘невязка’

Описание алгоритма:
Метод отражений основан на разложении матрицы A системы $Ax=b$ в произведение унитарной матрицы на верхнюю треугольную. Матрица A называется унитарной, если она удовлетворяет уравнению $A \cdot A^* = E$, где A^* - матрица, сопряженная с A. Вещественные унитарные матрицы называются ортогональными.

По своей структуре метод отражений близок к методу Гаусса, но исключение проводится с помощью матриц отражения, которые являются унитарными и эрмитовыми. Достоинством метода отражений является единая схема вычислительного процесса, не зависящая от структуры матрицы.

Теорема. Пусть S и l произвольные вектор-столбцы, причем вектор l имеет единичную длину. Тогда найдется такой вектор W, что построенная по нему матрица отражения $U = E - 2ww^*$ переведет вектор S в вектор, коллинеарный вектору l, т.е. $Us = \alpha l$.

Вектор W строится по правилу $w = \frac{1}{\rho}(s - \alpha l)$, где $|\alpha| = \sqrt{(\alpha,\alpha)}$, $\arg\alpha = \arg(s,l) - \pi$, $\rho = \sqrt{(s - \alpha l, s - \alpha l)} = \sqrt{2|\alpha|^2 + 2|\alpha|\,|(s,l)|}$.

Будем преобразовывать расширенную матрицу систему по правилу

$A_{k+1} = U_{k+1} A_k$, $k = 0,1,\ldots,n-2$

с помощью умножения слева на последовательность матриц отражения $U_1, U_2, \ldots, U_{n-1}$. Для построения матрицы U_1 на первом шаге метода в качестве вектора S берется первый столбец расширенной матрицы, а в качестве вектора I - координатный вектор $I = (1,0,0,\ldots,0)^T$. В силу выбора векторов S и I все координаты первого столбца расширенной матрицы, кроме первой, после выполнения первого шага метода будут равны нулю.

Пусть уже построена матрица A_k, у которой $a_{i,j}^{(k)} = 0$, $i > j$, $j = \overline{1,k}$. Теперь в качестве S и I берутся вектора

$$s = (0,\ldots,0,a_{k+1,k+1}^{(k)}, a_{k+2,k+1}^{(k)},\ldots,a_{n,k+1}^{(k)})^T,\ I = (0,\ldots,0,1,0,\ldots,0)^T,$$

где в векторе I единица стоит на $k+1$-ом месте. После выполнения k-го шага метода отражений получим матрицу A_{k+1}, у которой все элементы, стоящие ниже главной диагонали, в первых $k+1$-ом столбцах будут равны нулю. Невозможность выполнения очередного шага связана только с равенством нулю вектора S, а это невозможно, так как матрица A является невырожденной.

После $(n-1)$-шага получим матрицу, первые n столбцов которой образуют верхнюю треугольную матрицу L. Система уравнений, соответствующая полученной расширенной матрице, равносильна исходной системе. Значения неизвестных находятся аналогично обратному ходу метода Гаусса

$$x_n = -\frac{a_{n,n+1}^{(n-1)}}{a_{n,n}^{(n-1)}},\ x_i = -\frac{a_{i,n+1}^{(n-1)} + \sum_{j=i+1}^{n} a_{i,j}^{(n-1)} x_j}{a_{n,n}(n-1)}, i = n-1, n-2, \ldots, 1$$

Параллельная версия алгоритма подразумевает разделение матрицы A между процессами по столбцам. При этом на каждом этапе работы алгоритма один из процессов подсчитывает вектор *w,* и рассылает его остальным процессам. Получив нужный вектор, каждый процесс производит обновление всех столбцов своей части матрицы по правилу: *y_i -= 2*alpha*w, где alpha -* скалярное произведение w и y_i.

Во время параллельного выполнения обратного хода метода Гаусса процессы последовательно вычисляют переменные x_i, после чего отсылают остальным процессам вычеселнный x_y, а также вектор $x_i * y_i$

где y_i – i-й столбец верхнетреугольной матрицы.

Результаты выполнения.

Тестирование производилось на системе Blue Gene/P. Использовались матрицы размеров 1024 x 1024, 2048 x 2048, 4096 x 4096 и 8192x8192. Для 8192x8192 приведены результаты для 128, 256, 512 и 1024 процессов, так как на меньшем количестве процессов программа работает слишком долго.

Результаты:

Число процессов	Прямой ход	Обратный ход	Общее время	Невязка
Размер матрицы: 1024x1024				
1	14.378392	0.033966	14.412358	2.41936e-05
2	7.042283	0.041062	7.083345	2.34752e-05
4	3.51102	0.042798	3.553818	2.28543e-05
8	1.773241	0.04279	1.816031	2.3444e-05
16	0.901314	0.042516	0.94383	2.42236e-05
32	0.467877	0.042407	0.510284	2.52516e-05
64	0.250647	0.042455	0.293102	3.07107e-05
128	0.142551	0.04265	0.185201	3.78087e-05
256	0.081269	0.042645	0.123914	4.00912e-05
512	0.054555	0.043068	0.097623	3.728e-05
1024	0.04531	0.044474	0.089784	3.0256e-05
Размер матрицы: 2048x2048				
1	117.261658	0.13533	117.396988	0.000245255
2	56.995451	0.154057	57.149508	0.00023858
4	28.627211	0.162912	28.790123	0.00024121
8	14.474948	0.162518	14.637466	0.000238571
16	7.245684	0.162236	7.40792	0.00024104
32	3.679046	0.162154	3.8412	0.000246779
64	1.893055	0.163204	2.056259	0.000244394
128	1.008555	0.162996	1.171551	0.000244102
256	0.563857	0.162759	0.726616	0.000242769
512	0.346793	0.163547	0.51034	0.000237542
1024	0.23062	0.166299	0.396919	0.000239127

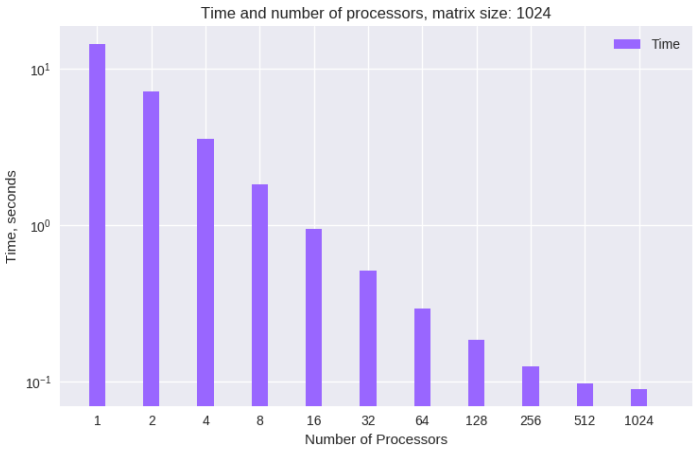
Число процессов	Прямой ход	Обратный ход	Общее время	Невязка
Размер матрицы: 4096x4096				
1	970.895762	0.548853	971.444615	0.000887317
2	499.833065	0.603065	500.43613	0.000893161
4	245.303601	0.613199	245.9168	0.000942643
8	123.787088	0.652621	124.439709	0.00110633
16	63.311317	0.648508	63.959825	0.0014199
32	31.129658	0.648449	31.778107	0.00165087
64	15.966749	0.649305	16.616054	0.00286529
128	7.847079	0.650521	8.4976	0.00326259
256	4.162589	0.649029	4.811618	0.00313611
512	2.323092	0.651341	2.974433	0.00296541
1024	1.41238	0.656122	2.068502	0.00264716
Размер матрицы: 8192x8192				
128	74.442184	2.553921	76.996105	0.00054045
256	38.216016	2.548331	40.764347	0.000541823
512	18.075943	2.545382	20.621325	0.000561798
1024	9.996802	2.556077	12.552879	0.000575152

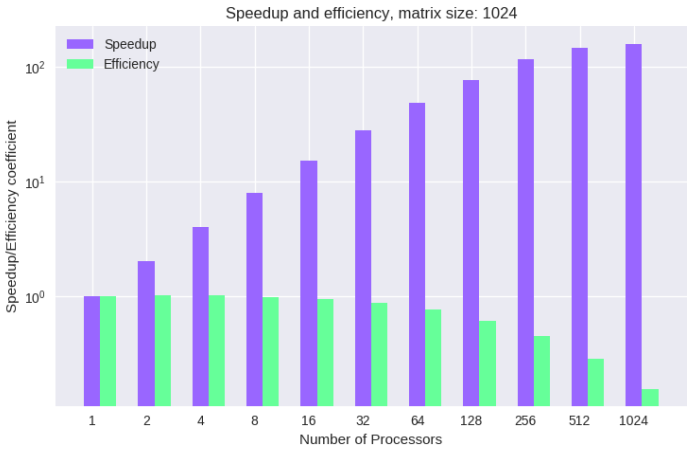
Ускорение и эффективность:

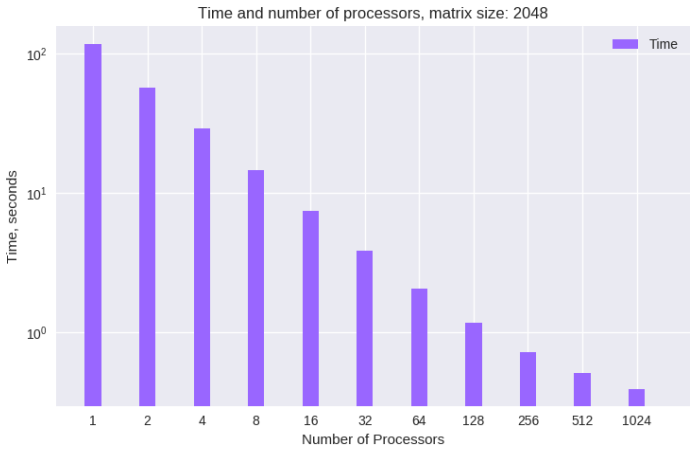
Число процессов	Ускорение	Эффективность	Ускорение	Эффективность
	1024x1024		**2048x2048**	
1	1.000	1.000	1.000	1.000
2	2.035	1.017	2.054	1.027
4	4.055	1.014	4.078	1.019
8	7.936	0.992	8.020	1.003
16	15.270	0.954	15.847	0.990
32	28.244	0.883	30.563	0.955
64	49.172	0.768	57.093	0.892
128	77.820	0.608	100.206	0.783
256	116.309	0.454	161.567	0.631
512	147.633	0.288	230.037	0.449
1024	160.523	0.157	295.771	0.289
	4096x4096		**8192x8192**	
1	1.000	1.000	-	-
2	1.941	0.971	-	-
4	3.950	0.988	-	-
8	7.807	0.976	-	-
16	15.188	0.949	-	-
32	30.570	0.955	-	-
64	58.464	0.914	-	-
128	114.320	0.893	-	-
256	201.896	0.789	-	-
512	326.598	0.638	-	-
1024	469.637	0.459	-	-

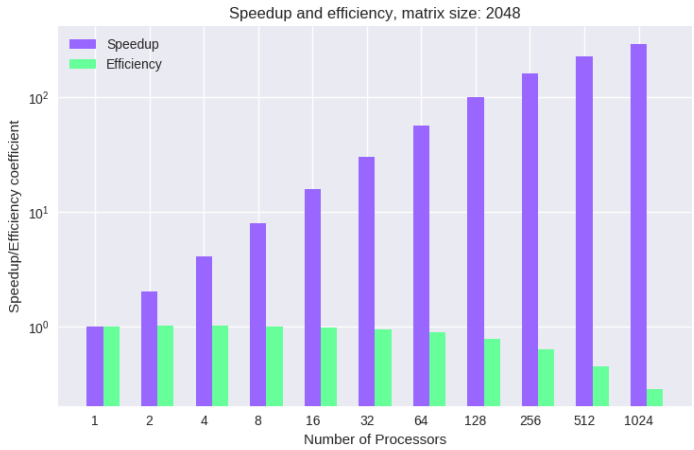
Графики

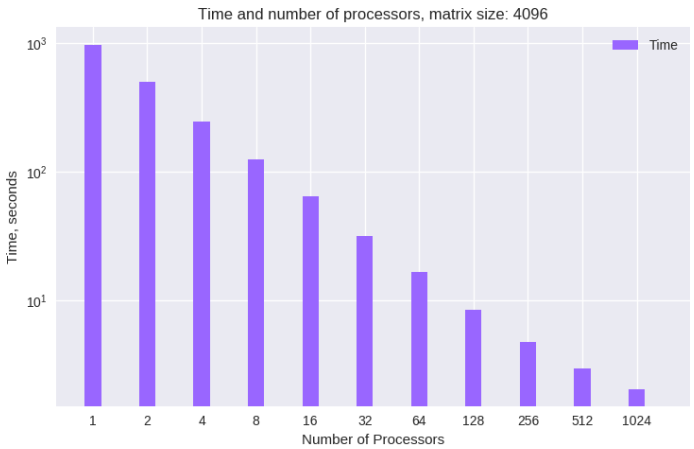
Далее приедены графики для матриц 1024x1024, 2048x2048, 4096x4096 и 8192x8192.

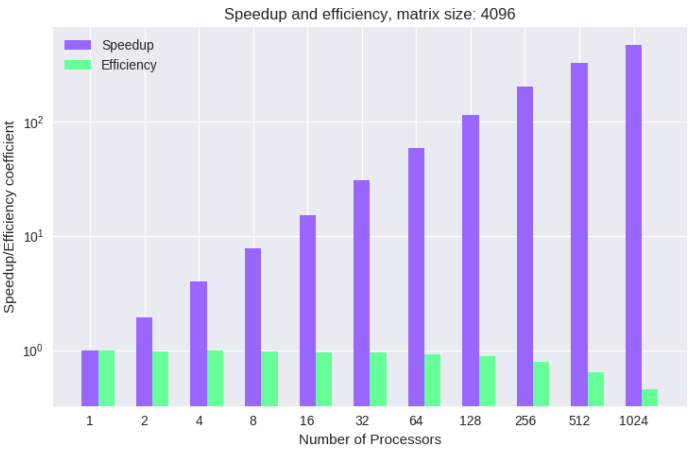


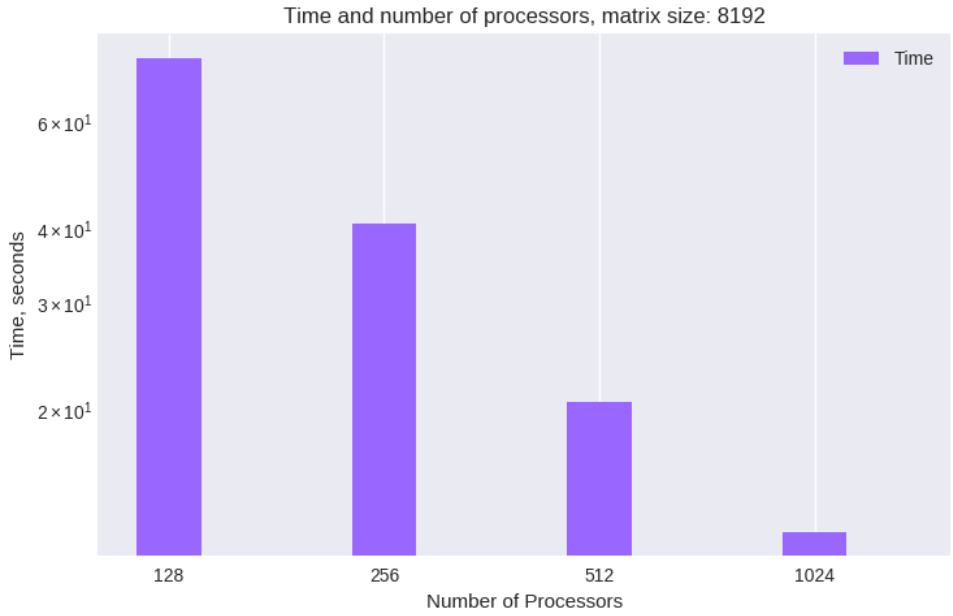












Основные выводы

С увеличением числа процессов время выполнения значительно уменьшается. Задача решения СЛАУ с помощью метода отражений эффективно распараллеливается, и можно получить значительное ускорение при достаточно высокой эффективности. При этом обратный ход метода Гаусса распараллеливается существенно хуже, чем приведение матрицы к верхнетреугольному виду.